

# Experimental Investigation of Control Updating Period Monitoring In Industrial PLC-based Fast MPC: Application to The Constrained Control of a Cryogenic Refrigerator

François Bonne<sup>1</sup>, Mazen Alami<sup>2</sup> and Patrick Bonnay<sup>1</sup>

<sup>1</sup> Univ. Grenoble Alpes, INAC-SBT, F-38000 Grenoble, France CEA, INAC-SBT, F-38000 Grenoble, France

<sup>2</sup> CNRS, Control Systems Department, University of Grenoble,  
11 rue des Mathématiques, Domaine Universitaire, 38402 Saint-Martin d'Hères, France

In this paper, a complete industrial validation of a recently published scheme for on-line adaptation of the control updating period in Model Predictive Control is proposed. The industrial process that serves in the validation is a cryogenic refrigerator that is used to cool the supra-conductors involved in particle accelerators or experimental nuclear reactors. Two recently predicted features are validated: the first states that it is sometimes better to use less efficient (per iteration) optimizer if the lack of efficiency is over-compensated by an increase in the updating control frequency. The second is that for a given solver, it is worth monitoring the control updating period based on the on-line measured behavior of the cost function.

*Index Terms*—Fast MPC, control updating period, real-time implementation, PLC-based MPC, cryogenic refrigerators.

## I. INTRODUCTION

Model Predictive Control (MPC) is an attractive control design methodology because it offers a natural way to express optimal objective while handling constraints on both state and control variables [17]. MPC design is based on the repetitive on-line solution of finite-horizon open-loop optimal control problems that are parametrized by the state value. Once the optimal sequence of control inputs is obtained, the first control in the sequence is applied over some updating period  $\tau_u$  during which, the new problem (based on the next predicted state) is solved and the resulted solution is applied while the prediction horizon is shifted by  $\tau_u$  time units and the process is repeated yielding an implicit state feedback.

The attractive features of MPC triggered attempts to apply it to increasingly fast systems. For such systems, the need for a high updating rate (small  $\tau_u$ ) may be incompatible with a complete solution of the underlying optimization problem during a single updating period  $\tau_u$ . This fact fired a rich and still active research area that is shortly referred to by "Fast MPC" (see [1, 12, 20, 13] and the reference therein).

Typical issues that are addressed in fast MPC literature concern the derivation of efficient computation of updating steps, reduction of the feedback delay, more or less rigorous computation of the Hessian, etc. Typical proofs of closed-loop stability in that context (see for instance [13]) depend on strong assumptions such as the proximity to the optimal solution, the quality of the Hessian matrix estimation, etc. With such assumptions, the corresponding stability proofs take the form of tautological assertions. In other words, when such assumptions are satisfied, the paradigm of fast MPC is

less relevant since standard execution of efficient optimizers would anyway give satisfactory results.

When the effectively applied control is far from being optimal (which is the case for instance after a sudden change in the set-point value) the hot-start (initialization of the decision variable after horizon shift) it induces for the next horizon does not necessarily decrease the cost function before several iterations. This is because far from the ideal solution, the final stabilizing constraints invoked in the formal proof of [17] may be far from being satisfied. On the other hand, if a constant large control updating period is used in order to accommodate for such situations, the overall closed-loop performance would be badly affected.

In recent papers [2, 3], investigations have been conducted regarding the impact of the choice of the control updating period  $\tau_u$  on the behavior of the cost function. Simple algorithms have been also proposed to monitor on-line the updating period based on the on-line behavior of the cost function to be decreased. More recently [4], it has been shown that the control updating period choice is intimately linked to the basic iteration being used. The two major facts that come out from these investigations can be summarized as follows:

**(Fact 1)** In a constant updating period schemes, it could be interesting to use less efficient (per iteration) algorithms provided that a significantly shorter updating period can be used [4]. This fact enhanced the recent interest [6, 16] in fast gradient-like algorithms [18] as a simpler approach when compared to second order algorithms. The work in [4] gives a formal explanation for this intuitively accepted fact.

**(Fact 2)** For a given optimization algorithm, the closed-loop

performance can be enhanced by an almost computational-free on-line adaptation rule of the control updating period [2, 3].

Obviously, a combination of the preceding facts holds also, namely, in adaptive frameworks, it can be more efficient to use simpler optimization algorithms provided that the gain induced by a higher updating rate compensates for the lack of efficiency per iteration.

In view of the preceding discussion, the contribution of this paper is twofold:

**First contribution.** This paper gives the first industrial validation of the proposed on-line adaptation of the control updating period. The realistic PLC-based implementation framework being used enhances the sensitivity of the closed-loop performance to the adaptation mechanism since it is several orders of magnitude slower than nowadays desk computers. As such, this paper gives a complete and realistic layout to understand the chain of concepts and methods that underline fast MPC paradigm.

**Second contribution.** Although simulation-based assessments have been proposed for Facts 1 and 2 mentioned above, these simulations always used first order gradient-based algorithms. Some promoters of second order algorithms may conjecture that such adaptation would be of no benefit since a second order scheme hardly needs more than a single iteration. This paper invalidates this conjecture by showing that 1) as far as the application at hand is concerned a first-order-like algorithm slightly outperforms a second order algorithm (in the realistic industrial hardware configuration at hand) strengthening Fact 1 in a constant updating period context. 2) the closed-loop performance of this first order algorithm can be improved by on-line adaptation of the control updating period. These two results put together infer that on-line adaptation is worth using even for second order algorithms and that a single iteration is not always sufficient for second order methods in realistic situations.

This paper is organized as follows: First, the problem is stated in section II by recalling the fast MPC implementation scheme and the main results of [3, 4]. In section III, the two algorithms that are used in the validation section are presented which are the QPOASES solver [14] and an Ordinary-Differential-Equation (ODE)-based solver that is briefly presented and then applied in the experimental validation. This second algorithm can be viewed as a first-order algorithm since it is based on the definition of an ODE in which the vector field is linked to the steepest descent direction. In section IV, the process is described, the control problem is stated and the computational PLC used in the implementation of the real-time MPC is presented in order to underline the computational limitation that qualifies the underlying problem as a fast MPC problem. The main contribution of the paper is given in section V, namely, extensive simulations are first given using the two above cited algorithms and using different constant control updating periods in order to investigate the

first fact discussed above. It is in particular shown that for both solvers, the locally (in time) optimal updating period changes dynamically depending on the context. Moreover, in order to draw conclusions that go beyond the specific case of the PLC at hand, several simulations are conducted using different conjectures regarding the PLC performances. This investigation shows that for the rather performant PLC we actually have today, the first order algorithm gives slightly better results, however, if faster future PLCs were to be used, QPOASES would give better results. This is the core message of the paper: the fast MPC paradigm is a matter of combined optimal choices involving the process bandwidth, the optimization algorithm, the available computational device, the control parametrization, etc. Finally, experimental results are shown under adaptive updating period. Section VII concludes the paper and gives hint for further investigation.

## II. BACKGROUND

### A. Definitions and Notation

Consider a general nonlinear system with state vector  $\mathbf{x} \in \mathbb{R}^n$  and a control vector  $\mathbf{u} \in \mathbb{R}^{n_u}$ . We consider a basic sampling period  $\tau > 0$  used to define the piece-wise constant (pwc) control profiles (a sequence of control values in  $\mathbb{R}^{n_u}$  that are maintained constant during  $\tau$  time units). As far as the general presentation of concepts is concerned, the general control parametrization is adopted according to which the whole control sequence is defined by a vector of decision variables  $p \in \mathbb{R}^{n_p}$  by:

$$\mathcal{U}_{pwc}(p) := (u^{(1)}(p) \quad \dots \quad u^{(N)}(p)) \in \mathbb{U} \subset \mathbb{R}^{Nn_u} \quad (1)$$

where  $u^{(i)}(p) \in \mathbb{R}^{n_u}$  is the control to be applied during the future  $i$ -th sampling period of length  $\tau$  while  $\mathbb{U} \subset \mathbb{R}^{n_u}$  is some admissible set. At this stage, no specific form is required for the system equations describing the dynamic model. The state  $\mathbf{x}_{k+j}$  that is reached - according to the model - after  $j$  sampling periods, starting from some initial state  $\mathbf{x}_k$ , under the sequence of control inputs  $\mathcal{U}_{pwc}(p)$  and some predicted disturbance  $\hat{\mathbf{w}}_k \in \mathbb{R}^{N \cdot n_w}$  is given by:

$$\forall j \in \{1, \dots, N\} \quad \mathbf{x}_{k+j} =: X(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k) \quad (2)$$

while the real state that is reached in the presence of true disturbances and/or model mismatched  $\tilde{\mathbf{w}}_k$  (that takes place over the time interval  $[k\tau, (N+j)\tau]$ ) is denoted by

$$X^r(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k, \tilde{\mathbf{w}}_k) \quad (3)$$

In the sequel, explicit mentioning of  $\tilde{\mathbf{w}}$  is sometimes omitted and the real state evolution is simply denoted by  $X^r(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k)$ .

It is assumed that an MPC strategy is defined by the following optimization problem that depends on the current state  $\mathbf{x}$  according to:

$$\mathcal{P}(\mathbf{x}) : \min_{p \in \mathbb{P}} J_0(p, \mathbf{x}) \quad \text{under } g(p, \mathbf{x}) \leq 0 \quad (4)$$

where  $\mathbb{P} \subset \mathbb{R}^{n_p}$  is the admissible parameter set,  $J_0$  is the cost function to be minimized while  $g(p, \mathbf{x}) \in \mathbb{R}^{n_c}$  defines

the set of inequality constraints.

Recall that in ideal MPC, the solution to (4), say  $p^{opt}(\mathbf{x})$  is used to define the feedback

$$K_{MPC}(\mathbf{x}) := u^{(1)}(p^{opt}(\mathbf{x})) \quad (5)$$

Indeed, ideal MPC frameworks assume that the optimal solution  $p^{opt}(\mathbf{x})$  is instantaneously available. In reality, the optimization problem  $\mathcal{P}(\mathbf{x})$  is solved using an iterative solver that is denoted by:

$$p^{(q)} = \mathcal{S}^{(q)}(p^{(0)}, \mathbf{x}) \quad (6)$$

where  $p^{(0)}$  stands for the initial guess while  $p^{(q)}$  is the iterate that is delivered after  $q$  successive iterations. In the sequel, the term *iteration* refers to the unbreakable set of operations (relative to  $\mathcal{S}$ ) that is necessary to deliver an update of  $p$ . In other words, if the time needed to perform a single iteration of  $\mathcal{S}$  on a given platform is denoted by  $\tau_1^{\mathcal{S}} > 0$ , then no update can be given in less than  $\tau_1^{\mathcal{S}}$  time units. Based on this remark, it seems reasonable to adopt updating instants that are separated by multiples of  $\tau_1^{\mathcal{S}}$ , namely:

$$t_{k+1} = t_k + q(t_k) \cdot \tau_1^{\mathcal{S}} \quad \text{with} \quad q(t_k) \in \mathbb{N} \quad (7)$$

where the  $t_k$ s are the instants where updated values of  $p$  can be delivered for use in the feedback control input. Moreover, we assume for simplicity that the basic sampling period  $\tau$  involved in the definition of the control parametrization map  $\mathcal{U}(p)$  is precisely  $\tau_1^{\mathcal{S}}$ , namely:

$$\tau = \tau_1^{\mathcal{S}} \quad (8)$$

Note that thanks to the flexibility of the parametrization, one can define pwc control profiles in which the control is maintained constant over multiples of  $\tau_1^{\mathcal{S}}$  while meeting (8) so that the latter condition is not really restrictive while it simplifies the description of the implementation framework.

Using the notation above, the real-life implementation scheme is defined as follows:

- (1)  $i \leftarrow 0$ ,  $t_i \leftarrow 0$ , some initial parameter vector  $p(t_0)$  is chosen. An initial number of iterations  $q(t_0) = q_0 \leq N$  is adopted.
- (2) The first  $q(t_i)$  elements of the control sequence  $\mathcal{U}(p(t_i))$  are applied over the time interval  $[t_i, t_{i+1} = t_i + q(t_i)\tau]$ .
- (3) Meanwhile, the computation unit performs the following tasks during  $[t_i, t_{i+1}]$ :
  - (3.1) Predict the future state  $\hat{\mathbf{x}}(t_{i+1})$  using the model and under the above mentioned sequence of controls. The time needed to achieve this very short time ahead prediction is assumed to be negligible for simplicity.
  - (3.2) Perform  $q(t_i)$  iterations to get

$$p(t_{i+1}) := \mathcal{S}^{(q(t_i))}(p^+(t_i), \hat{\mathbf{x}}(t_{i+1}))$$

where the initial guess  $p^+(t_i)$  is either equal to  $p(t_i)$  [cold start] or equal to some warm start that is derived from  $p(t_i)$  by standard translation technique.

- (4) At the updating instant  $t_{i+1}$  compute the number  $q(t_{i+1})$  of iterations to be performed during the next updating period  $[t_{i+1}, t_{i+2} = t_{i+1} + q(t_{i+1})\tau]$  using Algorithm 1 that is recalled in section II-B. As it has been shown in [3] and recalled hereafter, this updating costs no more than a dozen of elementary operations and can therefore be considered as instantaneous.
- (5)  $i \leftarrow i + 1$ , Goto step (2).

In the next section, the updating rule for  $q(t_{i+1})$  invoked in Step (4) of the implementation scheme is recalled. Note that by adapting  $q(t_i)$ , the control updating period  $\tau_u = q(t_i) \cdot \tau$  is adapted.

### B. Adaptation of the control updating period for a given solver $\mathcal{S}$

The following definition specifies a class of solvers that is invoked in the sequel and for which the adaptation mechanism recalled in this section can be applied:

**Definition 1:** A solver  $\mathcal{S}$  is said to be monotonic w.r.t the cost function  $J : \mathbb{R}^{n_p} \times \mathbb{R}^n \rightarrow \mathbb{R}$  if for all  $\mathbf{x}$ , the iterations defined by (6) satisfies:

$$J(p^{(i)}, \mathbf{x}) \leq J(p^{(i-1)}, \mathbf{x}) \quad (9)$$

for all  $i$ . This function is called hereafter the augmented cost function.  $\diamond$

Note that  $J$  generally differs from  $J_0$  involved in (4) because of the presence of constraints. A typical example of such  $J$  is given by the norm of the nonlinear function that gathers the Karush-Kuhn-Tucker (KKT) necessary conditions of optimality and when the solver uses a descent approach such as projected gradient or a specific implementation of Sequential Quadratic Programming (SQP) approach with trust region mechanism. Interior point-based algorithm can also enter in this category under certain circumstances in which the map  $J$  would be given by the penalized version of  $J_0$  involving barrier functions.

**Remark 1:** The conditions of Definition 1 can be relaxed in the following sense: if a solver  $\mathcal{S}$  satisfies the following condition:

$$J(p^{(i+\ell-1)}, \mathbf{x}) \leq J(p^{(i-1)}, \mathbf{x}) \quad (10)$$

for some map  $J$ , then the solver  $\mathcal{S}'$  that is derived from  $\mathcal{S}$  by:

$$\mathcal{S}'(p, \mathbf{x}) := \mathcal{S}^{(\ell)}(p, \mathbf{x}) \quad (11)$$

is monotonic in the sense of Definition 1 at the price of having a *single* iteration that takes  $\ell$ -times longer than  $\mathcal{S}$ , namely  $\tau_1^{\mathcal{S}'} = \ell \cdot \tau_1^{\mathcal{S}}$ .  $\diamond$

The following assumption is needed for the updating algorithm that can be stated as follows:

**Assumption 1:** The solver  $\mathcal{S}$  is monotonic and the corresponding map  $J$  [see Definition 1] is bounded below by a

strictly positive real  $\underline{J}$ , namely:

$$\forall(p, \mathbf{x}) \quad J(p, \mathbf{x}) \geq \underline{J} > 0 \quad (12)$$

Note that the last condition (12) can always be satisfied by adding an appropriate positive constant to the original cost.

In order to recall the updating algorithm proposed in [3], the following notations are needed:

$$J_k^+ := J(p^+(t_k), \hat{\mathbf{x}}(t_{k+1}))$$

the cost function value for the initial hot start  $p^+(t_k)$  (before any iteration is performed) and based on the predicted state at the future updating instant  $t_{k+1} = t_k + q(t_k) \cdot \tau$ .

$$\hat{J}_{k+1} := J(p(t_{k+1}), \hat{\mathbf{x}}(t_{k+1}))$$

the cost function value for the delivered value  $p(t_{k+1})$  (after  $q(t_k)$  iterations) and based on the predicted state at the future updating instant  $t_{k+1} = t_k + q(t_k) \cdot \tau$ .

$$J_{k+1} := J(p(t_{k+1}), \mathbf{x}(t_{k+1}))$$

the effectively obtained cost function value for the delivered value  $p(t_{k+1})$  and for the true state  $\mathbf{x}(t_{k+1})$  that is reached at instant  $t_{k+1} = t_k + q(t_k) \cdot \tau$ .

Based on these definitions, it comes out that the decrease of the augmented cost function can be studied by analyzing the behavior of the ratio  $J_{k+1}/J_k$  which can be decomposed according to:

$$\frac{J_{k+1}}{J_k} = E_k^r(q(t_k)) \times D_k^r(q(t_k)) \quad (13)$$

where

$$E_k^r(q(t_k)) := \frac{\hat{J}_{k+1}}{J_k^+} \quad ; \quad D_k^r(q(t_k)) := \frac{J_{k+1}}{\hat{J}_{k+1}} \times \frac{J_k^+}{J_k} \quad (14)$$

A deep analysis of the above terms shows that  $E_k^r(q)$  is linked to the current efficiency of the solver since it represents the ratio between the value of the augmented cost for the same predicted value  $\hat{\mathbf{x}}(t_{k+1})$  of the state before and after  $q(t_k)$  iterations are performed. The first ratio  $J_{k+1}/\hat{J}_{k+1}$  in  $D_k^r$  is 1 if the model is perfect since it represents the ratio between two values of the augmented function for the same value  $p(t_{k+1})$  of the parameter but for two different values  $\hat{\mathbf{x}}(t_{k+1})$  and  $\mathbf{x}(t_{k+1})$ . Finally, the ratio  $J_k^+/J_k$  is linked to the quality of the hot start since it represents the predicted ratio between two values of the augmented function before and just after the horizon shift.

The algorithm proposed in [3] recalled hereafter updates the number of iterations  $q(t_{k+1})$  to be performed during the next updating period so that the contraction ratio:

$$K_{k+1}^r(q(t_{k+1})) := E_{k+1}^r(q(t_{k+1})) \times D_{k+1}^r(q(t_{k+1})) \quad (15)$$

is lower than 1 and if this is achievable, the updating rule tries to minimize the corresponding expected response time  $t_r$

of the dynamics which is linked to the ratio  $q/\log(K_{k+1}^r(q))$ .

This leads to the following algorithm [3]:

---

**Algorithm 1** Updating rule for  $q(t_{k+1}^u)$

---

- 1: **Parameters**  $q_{max} \leq N$ ,  $\delta \in \{1, \dots, q_{max}\}$
  - 2: **Input data** (available after Step (3.2) page 3)
  - 3:  $q = q(t_k)$ ,  $p^{(0)} = p^+(t_k)$ ,  $p^{(i)} = \mathcal{S}^{(i)}(p^{(0)}, \hat{\mathbf{x}}(t_{k+1}))$
  - 4:  $J_k$ ,  $J_k^+$ ,  $\hat{J}_{k+1}$ ,  $J_{k+1}$
  - 5: Compute the following quantities:
 
$$E^r \leftarrow \hat{J}_{k+1}/J_k^+$$

$$D^r \leftarrow (J_{k+1}J_k^+)/(\hat{J}_{k+1}J_k)$$

$$K^r \leftarrow E^r \times D^r$$

$$\frac{\Delta D^r}{\Delta q} \leftarrow \frac{1}{q}[D^r - 1]$$

$$\frac{\Delta E^r}{\Delta q} \leftarrow \frac{J(p^{(q)}, \hat{\mathbf{x}}(t_{k+1})) - J(p^{(q-1)}, \hat{\mathbf{x}}(t_{k+1}))}{J(p^{(0)}, \hat{\mathbf{x}}(t_{k+1}))}$$

$$\frac{\Delta K^r}{\Delta q} \leftarrow E^r \cdot \left[ \frac{\Delta D^r}{\Delta q} \right] + D^r \cdot \left[ \frac{\Delta E^r}{\Delta q} \right]$$

$$\frac{\Delta t_r}{\Delta q} \leftarrow \frac{-\log(K^r) + \frac{q}{K^r} \times \frac{\Delta K^r}{\Delta q}}{[\log(K^r)]^2}$$
  - 6: **If**  $K^r \geq 1$  **then**  $\Gamma \leftarrow \frac{\Delta K^r}{\Delta q}$  **else**  $\Gamma \leftarrow \frac{\Delta t_r}{\Delta q}$
  - 7: **Output**  $q(t_{k+1}) \leftarrow \max\{2, \min\{q_{max}, q - \delta \cdot \text{sign}(\Gamma)\}\}$
- 

Roughly speaking, this algorithm implements a step of size  $\delta$  in the descent direction defined by the sign of the approximated gradient  $\Gamma$ . The step is projected into the admissible domain of  $q \in \{2, \dots, q_{max}\}$ . More details regarding this algorithm are available in [3].

Section V shows the efficiency of the proposed algorithm when applied to a given solver for the PLC-based implementation of MPC to the cryogenic refrigerator. Before this, the next section gives a simple argumentation that underlines a fundamental trade-off between the efficiency (per iteration) of a solver and the basic corresponding unbreakable computation time  $\tau_1^S$ . This is done in adaptation-free context in order to decouple the analysis.

### C. Fundamental trade-off in the choice of solvers

Let us consider a solver  $\mathcal{S}$  and the corresponding time  $\tau_1^S$  that is needed to perform the unbreakable amount of computations involved in a single iteration. Given a control updating period  $\tau_u$ , the number of iterations that can be performed is equal to  $q = \lfloor \tau_u / \tau_1^S \rfloor$  and the corresponding variation of the augmented cost function would be given by:

$$J_{k+1} - J_k := \underbrace{J_{k+1} - J_k^+}_{-E_k^S(\tau_u)} + \underbrace{J_k^+ - J_k}_{D_k(\tau_u)} \quad (16)$$

where here again,  $E_k^S(\tau_u)$  and  $D_k(\tau_u)$  are linked to the current efficiency of the solver ( $E_k^S$ ) and the combined

effect of model mismatch and the horizon shift effect on the cost function respectively. Both terms depend obviously on  $\tau_u$ . Indeed  $E_k^S(\tau_u)$  depends on  $\tau_u$  through the number of iterations while  $D_k(\tau_u)$  depends on  $\tau_u$  since when  $\tau_u = 0$  then  $D_k$  vanishes (no prediction error and no possible bad hot start). Note that  $E_k^S$  and  $D_k$  are absolute (non relative) versions of the relative maps  $E_k^r$  and  $D_k^r$  invoked in section II-B to introduce Algorithm 1. Note also that unlike the efficiency indicator  $E_k^S(\tau_u)$  which heavily depends on the solver, the  $D_k(\tau_u)$  term is solver-independent.

Figure 1 shows typical allures of these terms for two different solvers  $\mathcal{S}_1$  (most efficient) and  $\mathcal{S}_2$  (less efficient). It can be seen that the iterations of  $\mathcal{S}_1$  are more efficient at the price of longer computation time  $\tau_1^{\mathcal{S}_1} > \tau_1^{\mathcal{S}_2}$ . The dots on the right hand plot recall that the updating can be delivered only at quantized updating instants.

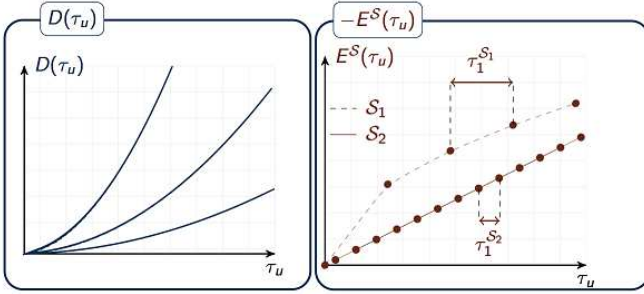


Fig. 1: Possible allures of the  $D_k(\tau_u)$  and  $E_k^S(\tau_u)$  in realistic fast NMPC implementations. The right figure shows the efficiency maps for two different solvers corresponding to two different computation times per iteration  $\tau_1^{\mathcal{S}_1}$  and  $\tau_1^{\mathcal{S}_2}$ .

Now based on (16), the decrease of  $J_k$  is conditioned by the inequality:

$$E_k^S(\tau_u) > D_k(\tau_u) \quad (17)$$

which expresses the need to have the  $E_k^S$  curves above the  $D_k$  curve for the adopted value of the updating period.

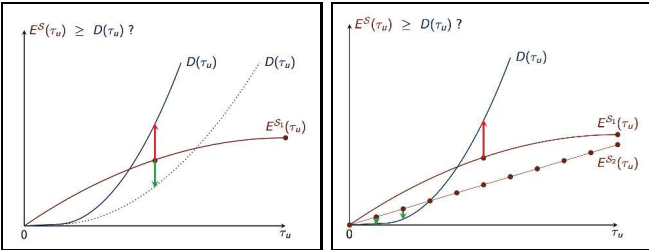


Fig. 2: (Left) Use of the most efficient solver  $\mathcal{S}_1$ : depending on the context, there are possible configurations of  $D$  that make the decrease of the augmented cost impossible. (Right) In such cases, the use of the less efficient solver  $\mathcal{S}_2$  enables to decrease the augmented cost thanks to shorter updating periods.

Figure 2 gives a qualitative illustration of the resulting fundamental trade-off: The left plots shows situations where the use of the more efficient solver  $\mathcal{S}_1$  makes (17) impossible

to satisfy whatever is the updating period being used. In such cases, the right plot shows that less efficient solvers like  $\mathcal{S}_2$  together with appropriate short updating periods can satisfy the decreasing condition (17). The right figure also shows that in this latter case, there may be several possible values of  $\tau_u$  (several number of iterations) that may decrease the cost and an adaptive on-line monitoring algorithm like the one recalled in section II-B may be appropriate to get closer to an optimal decrease.

In the following sections, the two solvers that are used in the validation section are introduced.

### III. PRESENTATION OF THE ALGORITHMS

#### A. qpOASES

The qpOASES [15] solver is a well know solver in the linear constrained MPC control community. It offers a very efficient implementation of the active-set strategy [14]. If several QP problems must be solved with constant Hessian and constraint matrices, the qpOASES package offers the possibility of hot-starting from previous solution with a subroutine called *qpOASES\_sequence*. In the sequel, the *qpOASES\_sequence* subroutine will be used and will simply be recalled as qpOASES.

#### B. ODE-based solver

In this section, an ODE-based solver that is used hereafter to implement the PLC-based constrained MPC is briefly presented. The real-time performance of this solver is also compared to that of qpOASES in the PLC constrained performance setting in order to illustrate Fact 1 mentioned above.

Consider the Quadratic Programming (QP) problem defined by:

$$\tilde{\mathcal{P}}(z) = \begin{cases} \min: J_0(z) = z^T \Phi z + z^T \phi \\ \text{under } \begin{cases} \Gamma z - \gamma \leq 0 \\ \underline{z} \leq z \leq \bar{z} \end{cases} \end{cases} \quad (18)$$

where  $z \in \mathbb{R}^{n_z}$  is the decision variable while  $\Phi$  and  $\phi$  are matrices of appropriate size.  $\Gamma \in \mathbb{R}^{n_c \times n_z}$  and  $\gamma \in \mathbb{R}^{n_c}$  are the matrices that define the set of  $n_c$  inequality constraints while  $\underline{z}$  and  $\bar{z}$  are lower and upper bounds on the decision variables.

Based on the above formulation, the following augmented cost function can be defined:

$$J(z) := J_0(z) + \alpha \sum_{i=1}^{n_c} \max(\Gamma_i z - \gamma_i, 0)^\mu + \alpha \sum_{i=1}^{n_z} \max(z_i - \bar{z}_i, 0)^\mu + \alpha \sum_{i=1}^{n_z} \max(\underline{z}_i - z_i, 0)^\mu \quad (19)$$

where  $\Gamma_i \in \mathbb{R}^{1 \times n_z}$  is the  $i$ -th line of  $\Gamma$ . Based on this augmented cost, the following Ordinary Differential Equation

(ODE) can be used to define a trajectory in the decision variable space along which the augmented cost decreases:

$$\dot{z} = -\frac{dJ}{dz}(z) \quad (20)$$

Note however that this ODE is generally stiff because of the high values of  $\alpha$  one needs to use in order to enforce the constraints fulfillment. That is the reason why the one-step Backward-Differentiation-Formulae (TR-BDF2) described in [5] for stiff differential equations is used here.

Note also that after the computation of the TR-BDF2 step, all the decision variable that correspond to hard constraints (saturation on actuator for instance) are projected into the admissible box before a next iteration is computed. In addition to the integration scheme described in [5], the initial time step is defined by using the following expression:

$$\Delta t = \sqrt{\frac{1}{\|\dot{z}(t)\|}} \quad (21)$$

In the case of the quadratic problem described in paragraph IV-C, this method leads to fast convergence to the suboptimal solution  $z^*$ , being very close to the actual optimal solution of the original problem even with real-time constraints. The comparison between solvers III-A and III-B will be done in paragraph V-A.

Note also that this solver fully satisfies the decrease condition (9) since it moves along the descent trajectory defined by (20). Therefore, the adaptation mechanism of the control updating period can be applied.

#### IV. PLANT DESCRIPTION

##### A. General presentation

Fig. 3 shows an overview of the cryogenic plant of the CEA-INAC-SBT, Grenoble. This plant provides a nominal cooling capacity of 450 W at 4.4 K in the configuration in which this study have been done. It is dedicated to physical experiments (cryogenic component testing, turbulence and pulsed heat load studies, etc.).

The process flow diagram of the cryogenic plant is shown in Fig. 4. One may notice the following main elements:

- Two volumetric screw compressors ( $NC_*$ ) and a set of control valves ( $CV_{95*}$ ),
- Several counterflow heat exchangers ( $NEF_*$ ), a liquid nitrogen pre-cooler ( $NEF_5$ ),
- A cold turbine expander which extracts work from the circulating gas ( $Stt_{207}$ ),
- A so-called turbine valve ( $CV_{156}$ ),
- A Joule-Thomson expansion valve for helium liquefaction ( $CV_{155}$ ),
- A phase separator ( $NS_1$ ), connected to the loads (simulated here by the heating device referred as  $NCR_{22}$ ).

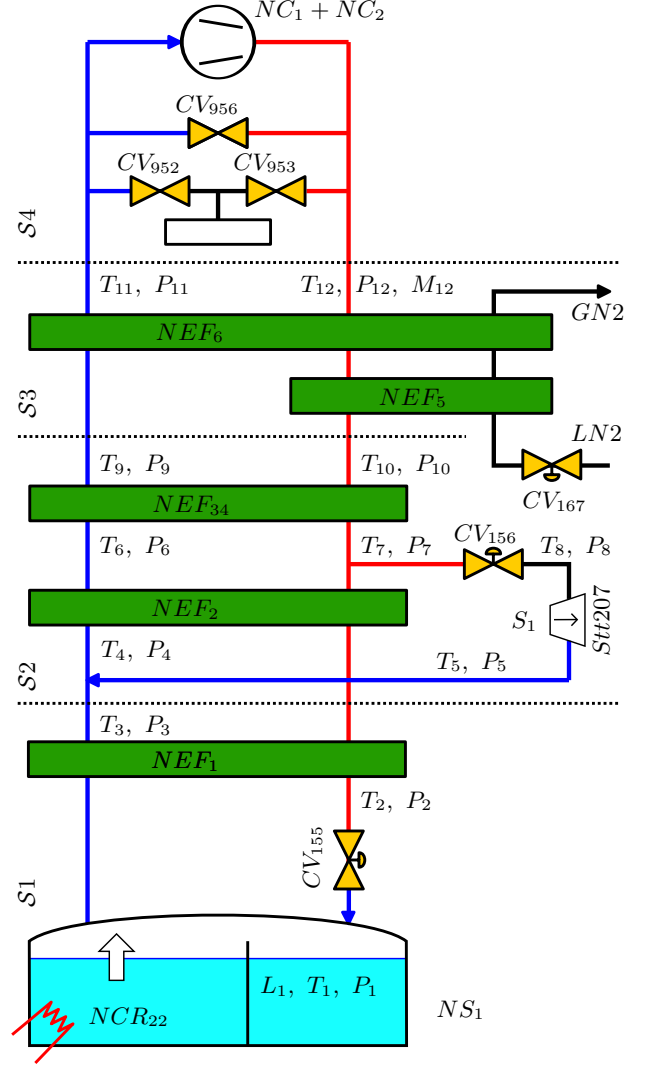


Fig. 4: Functional overview of the 450W at 4.4K helium refrigerator available at CEA-INAC-SBT, Grenoble. The components named  $CV$  are controlled valves, used to control the system. The label  $Stt$  stands for the cryogenic turbine while  $NS$  is used for the phase separator.  $NC$ 's are helium compressors while  $NEF$ 's stand for heat exchangers.  $T$ 's and  $P$ 's stand for temperature and pressure sensors.  $S_1$  is the turbine speed sensor while  $L_1$  stands for the bath level sensor.

Note that the plant can be viewed as the interconnection of four elementary subsystems: the Warm Compression Station ( $S_4$ ), the Nitrogen Pre-Cooler ( $S_3$ ), the Brayton Cycle ( $S_2$ ) and the JT cycle ( $S_1$ ), delimited by dotted lines in Fig. 4. While constrained MPC is used in this study, the cryogenic system is classically controlled by three independent control loops:

- The output temperature of the turbine expander is controlled with a PI controller working with the turbine valve  $CV_{156}$ ;
- the level of liquid helium in the tank is controlled by a PI controller, working with the heating device  $NCR_{22}$ ;
- the high and low pressures (in red and blue pipes,



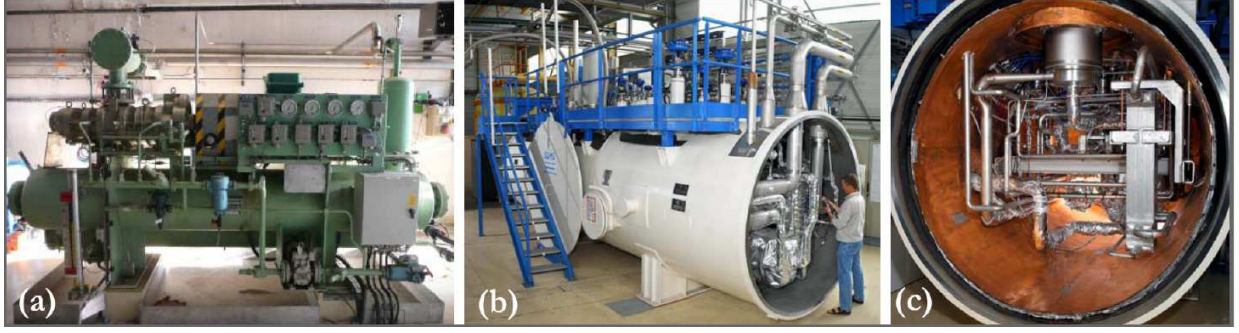


Fig. 3: Views of the cryogenic plant of CEA-INAC-SBT, Grenoble. (a) The screw compressor of the warm compression station. (b) The cold box. (c) Internal detail of the cold box.

respectively) is controlled by an LQ controller, like the one described in [8];

the valve  $CV_{155}$  being used at a constant opening set by the user, depending on the application. In this study, attention has been focused on subsystems 1 and 2, with are the coldest part of the refrigerator (from  $80K$  to  $4.4K$ ). More informations about the plant can be found in [10].

### B. Model derivation and properties

In order to derive the system model, several studies have been conducted [10, 11, 9, 7]. The Joule-Thompson cycle of this paper has been modelled in [9] while the Brayton cycle is presented in details in [7]. It is worth mentioning that heat exchangers involve models with coupled partial differential equations (PDEs) that have been spatially discretized, leading to rather large state space. In this study, the two models has been merged to obtain a state space model that takes the following form:

$$\dot{\mathbf{x}} = f^1(\mathbf{x}, \mathbf{u}, \mathbf{w}) \quad (22a)$$

$$\mathbf{y} = f^2(\mathbf{x}, \mathbf{u}, \mathbf{w}) \quad (22b)$$

where  $f^1$  is the function that express the derivative of the state  $\mathbf{x}$  while  $f^2$  is the function that express the measured output vector  $\mathbf{y}$ . Both functions are continuously differentiable. State vector, input vector, and disturbance vector are expressed more precisely by

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_{ns1} \\ \mathbf{x}_{nef2} \\ \mathbf{x}_{nef1} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} CV_{155} \\ NCR_{22}^A \\ CV_{156} \end{pmatrix}, \quad \mathbf{w} = NCR_{22}^{HL} \quad (23)$$

where  $\mathbf{x}_{ns1}$ ,  $\mathbf{x}_{nef1}$  and  $\mathbf{x}_{nef2}$  depict the state vector of individual components, described in [9, 7]. It has to be noted that  $NCR_{22}$  is used both to control the plant and to disturb it. That is why it has been named  $NCR_{22}^A$  for the actuator part and  $NCR_{22}^{HL}$  for the heat load part. At the end,  $NCR_{22} = NCR_{22}^{HL} + NCR_{22}^A$ . The vector of measured output is the following:

$$\mathbf{y} = (L_1 \quad V_1 \quad T_1 \quad \cdots \quad T_{10} \quad P_1 \quad \cdots \quad P_{10})^T \quad (24)$$

It has been shown in [7] that the non-linear model expressed by (27) can be linearized around an operation point of interest defined by  $f^1(\mathbf{x}_0, \mathbf{u}_0, \mathbf{w}_0) = 0$ . The linearized model is then

discretized using Matlab function  $c2d(\cdot)$  with sampling period  $\tau = 5s$ , leading to the following discrete LTI model:

$$\tilde{\mathbf{x}}_{k+1} = A\tilde{\mathbf{x}}_k + B\tilde{\mathbf{u}}_k + F\tilde{\mathbf{w}}_k \quad (25)$$

$$\tilde{\mathbf{y}} = C\tilde{\mathbf{x}}_k + D\tilde{\mathbf{u}}_k + G\tilde{\mathbf{w}}_k \quad (26)$$

where variables with a tilde depict the deviation of the original variables around the operating point of interest:

$$\mathbf{x}_k = \tilde{\mathbf{x}}_k + \mathbf{x}_0, \quad \tilde{\mathbf{u}}_k = \mathbf{u}_k - \mathbf{u}_0 \quad (27)$$

$$\mathbf{y}_k = \tilde{\mathbf{y}}_k + \mathbf{y}_0, \quad \tilde{\mathbf{w}}_k = \mathbf{w}_k - \mathbf{w}_0$$

Note that the model defined by (26) stands for the prediction model (2) invoked in the general presentation of MPC (section II-A). Following the same notation, the predicted output is denoted by  $\mathbf{y}_{k+j} = Y(j, \mathbf{x}_k, p, \tilde{\mathbf{w}}_k)$  while the true measured output is denoted by  $Y^r(j, \mathbf{x}_k, p, \tilde{\mathbf{w}}_k)$ .

### C. Statement of the MPC-related optimisation problem

First of all, the following constraints have to be satisfied as far as possible:

$$\underline{y}^c \leq \mathbf{y}_k^c \leq \overline{y}^c \quad (28a)$$

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \overline{\mathbf{u}} \quad (28b)$$

$$\underline{\delta \mathbf{u}} \leq \delta \mathbf{u}_k \leq \overline{\delta \mathbf{u}} \quad (28c)$$

where  $\delta \mathbf{u}_k$  stand for the increment  $\mathbf{u}_k - \mathbf{u}_{k-1}$  on the input vector.  $\mathbf{y}_k^c$  denotes a subset of output components  $\mathbf{y}_k$  which is constrained. This subset is composed of the helium bath level  $L_1$  and the turbine output temperature  $T_5$ . Details regarding the variables involved in (28) are given in table I:

Var.	Meaning	Value
$\underline{\mathbf{u}}$	min. control effort	$(20 \ 20 \ 0)^T$
$\overline{\mathbf{u}}$	max. control effort	$(60 \ 60 \ 150)^T$
$\underline{y}^c$	low limit on the output	$(59 \ 16)^T$
$\overline{y}^c$	high limit on the output	$(61 \ 9)^T$
$\underline{\delta \mathbf{u}}$	max increment	$(0.5 \ 10 \ 0.1)^T$
$\overline{\delta \mathbf{u}}$	min increment	$(0.5 \ 10 \ 0.1)^T$

TABLE I: The constraints bounds

One of the specific feature of Output constraints is that they cannot be necessarily fully respected depending on the unpredictable thermal loads. That is why these constrained

are systematically relaxed. This is introduced through the constraint violation variable  $\mathbf{v}_k$  that is defined as follows:

$$\mathbf{v}_k = \max(\mathbf{y}_k^c - \bar{\mathbf{y}}^c, 0) + \max(\underline{\mathbf{y}}^c - \mathbf{y}_k^c, 0) \quad (29)$$

while constraint violation prediction at sampling instant  $k + j$  is written:

$$V(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k) = \max(Y^c(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k) - \bar{\mathbf{y}}^c, 0) + \max(\underline{\mathbf{y}}^c - Y^c(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k), 0) \quad (30)$$

where  $Y^c(j, \mathbf{x}_k, p, \mathcal{W})$  is used to define the constrained subset of  $Y(j, \mathbf{x}_k, p, \mathcal{W})$ .

The sequence of control vectors  $u^{(i)}(p)$  is then obtained by minimizing the cost function :

$$J_k = \sum_{j=1}^{N_p} \|X(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k)\|_Q^2 + \|u^{(j)}(p)\|_R^2 + \|V(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k)\|_\rho^2 \quad (31)$$

where  $Q$  and  $R$  are weighting matrices on the state and input vectors while  $\rho$  defines the constraint violation-related penalty. This cost function, together with the linear constrained and the linearized model (26) lead to a constrained QP problem which is of the form (18) in which the decision variable  $z$  is precisely the control profile parameter  $p$ . Note also that the affine term  $\phi$  [see (18)] does depend on the current value of the disturbance  $\mathbf{w} = NCR_{22}^{HL}$ .

By choosing a sampling period  $\tau = 5$  sec, preliminary simulations showed that a prediction horizon of at least  $N_p = 100$  is required. This leads to an optimization problem that involves 700 decision variables and a total number of 1000 constraints to be satisfied if trivial pwc parametrization is adopted. Such problem are beyond the computational capacity of the targeted industrial PLC (see the performance of our PLC in the section IV-D).

To reduce the problem dimension, the control profile has been parametrized using classic piece-wise affine method that leaves as decision variables the values of the control inputs at 7 decisions instants<sup>1</sup>. Moreover, the constraints satisfaction is checked only at 14 future instants<sup>2</sup>. This finally leads to an optimization problem involving 49 decision variables (note that there are 7 control inputs, namely 3 physical input and 4 virtual input representing the constraints violation), with 56 (outputs) plus 38 rate saturation constraints to be satisfied.

To ensure that this scheme is appropriate to control the plant, the problem closed-loop system is first simulated using the qpOASES solver. Time results are presented in Fig. 5. Fig. 5 (a) shows the thermal heat load that has been used in this simulation. Part (b) shows that the scheme is able to decrease the stage instantaneous cost define as:

$$\tilde{J}_k^{inst} = \|\mathbf{x}_k^r\|_Q^2 + \|\mathbf{u}_k\|_R^2 + \|\mathbf{v}_k\|_\rho^2 \quad (32)$$

<sup>1</sup>decisions instants are chosen to be: (1, 2, 4, 8, 16, 50, 50, 100)

<sup>2</sup>constraints verifications instants are chosen to be:

(1, 2, 3, 4, 6, 8, 16, 24, 32, 48, 60, 72, 84, 100)

Parts (c) and (d) of Fig. 5 show that the constraints are violated within limited amplitude and duration. Part (e) shows the control effort. Part (f) shows the number of iterations of the qpOASES solver. It is worth mentioning that the number of iterations is important during heat load event. This has significant consequence on real-time feasibility of the qpOASES-based solution as it is examined in the sequel.

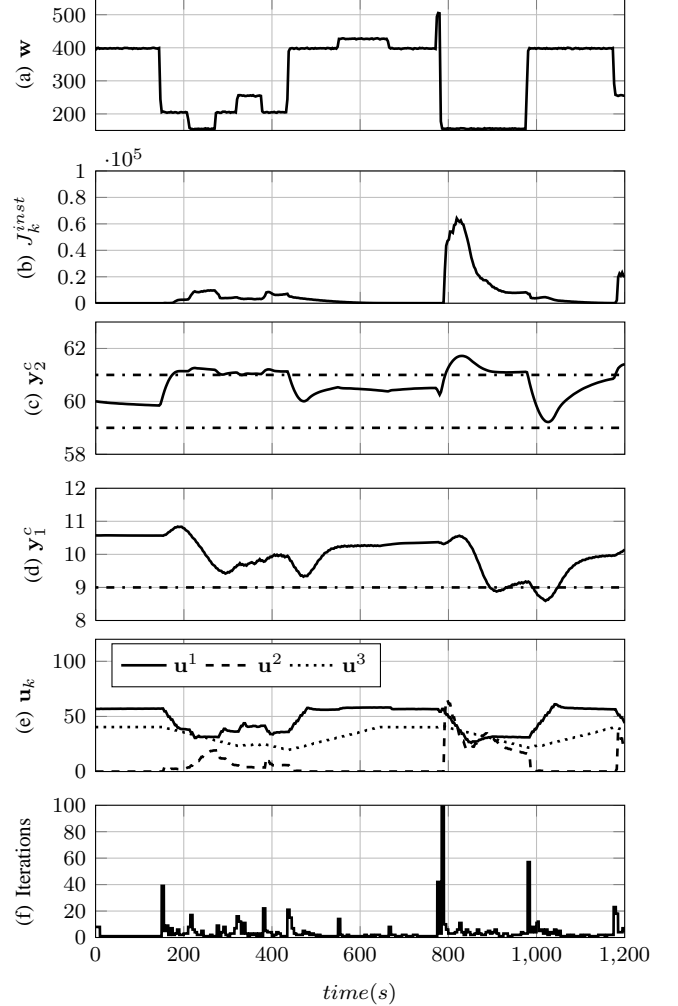


Fig. 5: Simulated behaviour of the system under qpOASES-based MPC control without limitation of the number of iterations.

#### D. Description of the PLC

This section focuses on the Programmable Logic Controller (PLC) available to implement the QP-based constrained MPC. It is a Schneider TSX P574634M shown in Fig. 6. This PLC is fully dedicated for our application and it communicates optimisation results to another PLC that actually controls the plant.



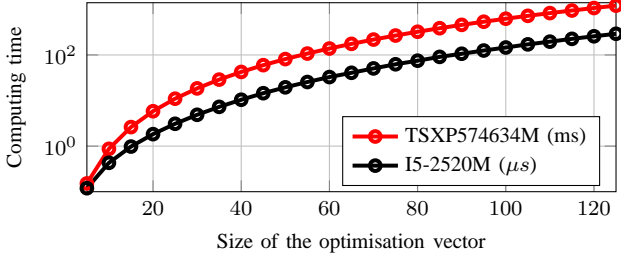


Fig. 7: Cholesky factorisation time for two different CPUs. It can be noticed that the performance ratio between the PLC and the laptop is about 4000 for matrices sized 40 to 125.



Fig. 6: Schneider PLC TSX P574634M

According to the manufacturer, this PLC shows maximum computing capability of about  $1.8 \text{ Mflops}$  [19]. In order to evaluate this claim, the Cholesky factorisation of increasing size matrices has been executed while monitored the computation times. Fig. 7 shows the results compares them to the performance of a nowadays DELL Latitude E6520 laptop with Intel I5-2520M CPU. This graph shows a slowing factor that lies around 4000. Note that the same graph shows the performance of the PLC in ms while the performance of the desk computer is shown in  $\mu s$ .

Note that the PLC is used with an external PCMCIA memory card of  $2 \text{ Mb}$ , shared for both code and variables. This makes memory also a crucial issue. Indeed without reduced parametrization, the Hessian of the QP problem would have just fit the memory size of the PLC, since it represents a total memory occupation  $4 * 700^2 = 1.96 \text{ Mb}$  in single precision representation.

Now since a single iteration of the qpOASES solver takes approximately  $120 \mu s$ , the same iteration would take  $0.12 * 4000 = 0.48 s$  when executed on the PLC. Therefore only 10 iterations of the qpOASES solver can be performed during the sampling period  $\tau = 5 \text{ sec}$ . The scenario that has been shown in Fig. 5 with no bound on the number of iterations has been simulated with the qpOASES 'maxiter' option set to 10. The result is presented by Fig. 8 on which the unlimited case has been also reported for easiness of comparison.

Figure 8 shows that when the number of iterations of the hot-started qpOASES solver is limited to 10, the closed-

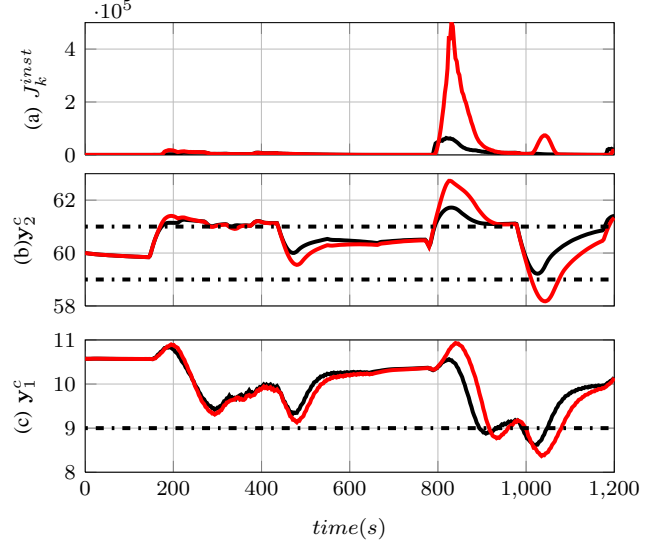


Fig. 8: Simulated behaviour of the system under MPC control for both unconstrained (black lines) and constrained (red lines) solving time

loop performance as well as the constraints fulfillment are drastically affected. This is precisely for this reason that the ODE-based solver explained in section III-B has been developed since it corresponds to a less computation time per iteration and can therefore be potentially more suitable in presence of the limited performance available PLC following the discussion of section II-C.

## V. FAST MPC-RELATED INVESTIGATION

### A. Comparison of algorithms

The aim of the present section is to assess the first Fact mentioned above, namely that it is sometimes better to use a less efficient per iteration solver (the ODE-based solver in our case) provided that it corresponds to less computation time per iteration. In our case, as far as the above described PLC is used, it is possible to perform 20 iterations of the ODE-based solver against only 10 iterations of the qpOASES solver.

Eight hours simulations have been done with the two solvers, with a variable computational capability (i.e. a variable allowed number of iterations). Some relevant results are plotted, always as a function of the normalized computation capability  $\bar{P} = P/P_0$  where  $P_0$  is the computational capability of our device.

In order to support the comparison that can be difficult because of the presence of relaxed weighted constraints, the cost (31) to be minimized at each sampling period has been divided in two separated parts, in order to compare them separately. The first part represents the deviation cost:

$$\bar{J}_k^{dev} = \sum_{j=1}^{N_p} \|X(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k)\|_Q^2 + \|u^{(j)}(p)\|_R^2 \quad (33)$$

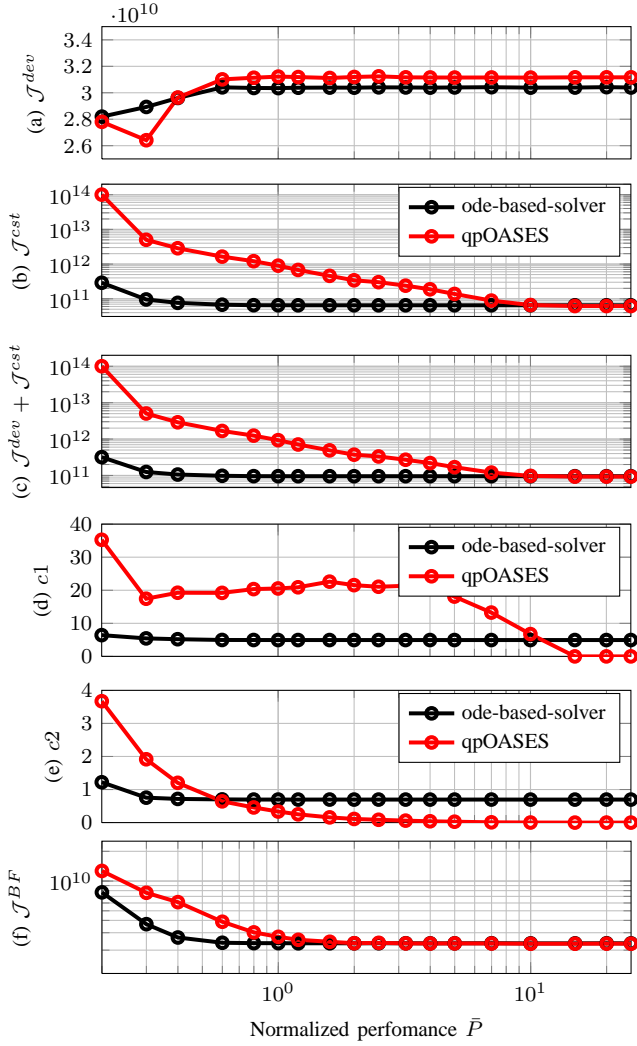


Fig. 9: Performance indicators of the two solvers comparison vs the normalized computation power. The case  $\bar{P} = 1$  corresponds to the PLC we dispose of and which is presented in section IV-D.

while the second part stands for the outputs constraints violation cost:

$$\bar{J}_k^{cst} = \sum_{j=1}^{N_p} \|V(j, \mathbf{x}_k, p, \hat{\mathbf{w}}_k)\|_\rho^2 \quad (34)$$

and then the sum of those two costs along the whole simulation is expressed:

$$\bar{J}^{dev} = \sum_{k=1}^{N_{sim}} \bar{J}_k^{dev} \quad (35) \quad \bar{J}^{cst} = \sum_{k=1}^{N_{sim}} \bar{J}_k^{cst} \quad (36)$$

where  $N_{sim}$  is the number of problems solved during the simulation.

Then, constraints respect is presented in two different manners:

$$c_1 = \max_{k \in \{1, \dots, N_{sim}\}} \max_{j \in \{1, \dots, n_c\}} \max\{\Gamma_j p_k - \gamma_j, 0\} \quad (37)$$

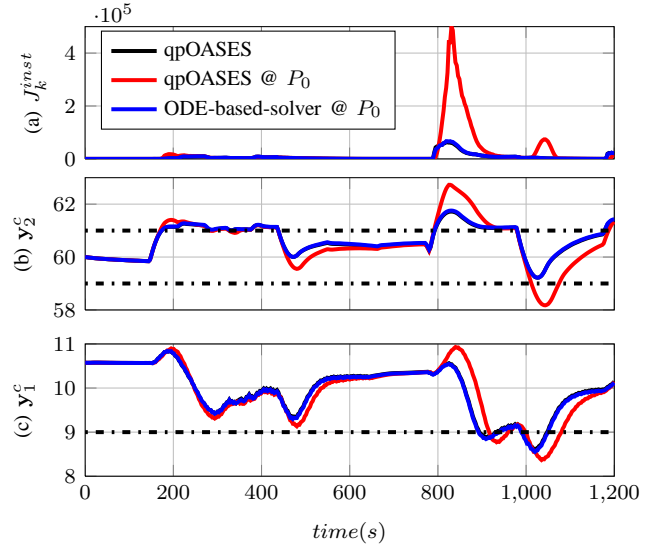


Fig. 10: Comparison of the closed-loop performance under the ODE-based solver (20 iterations), the qpOASES solver (10 iterations) and the qpOASES (without limitations).

being the maximum predicted constraints violation during the simulation while

$$c_2 = \frac{1}{N_{sim}} \sum_{k=1}^{N_{sim}} \max_{j \in \{1, \dots, n_c\}} \max\{\Gamma_j p_k - \gamma_j, 0\} \quad (38)$$

being the average predicted constraints violation during the simulation.

Finally, a closed-loop cost has been calculated according to:

$$\bar{J}^{BF} = \sum_{k=0}^{N_{sim}} \bar{J}_k^{inst} \quad (39)$$

The quantities (35), (36), (35)+(36), (37), (38) and (39) are shown in Fig. 9 against normalized computational performance  $\bar{P}$ . It can be noticed that the suboptimal ODE-based solver is behaving better than qpOASES in the case of low performance computation devices, while the qpOASES solver becomes clearly better beyond some hardware performance indicator. The trajectories of the two closed-loop results are shown in Fig. 10, comparing the two solvers for the nominal PLC performance  $P_0$  against the result obtained with the qpOASES solver with limited number of iterations and with the 10 maximum number of iterations. It comes clearly that the use of the less efficient (per iteration) solver with 20 iterations outperform the use of 10 iterations of the qpOASES solver. Moreover, the use of the ODE-based solver enables the nominal qpOASES (without limitation) performance to be recovered.

### B. Control updating period monitoring

In the section, attention is focused on the ODE-based solver. First of all, simulations will be done for updating period from one to five (i.e. a number of iterations from 4 to 20), and it will be shown that quadratic performances vary and there is an optimum to be found. Then the algorithm

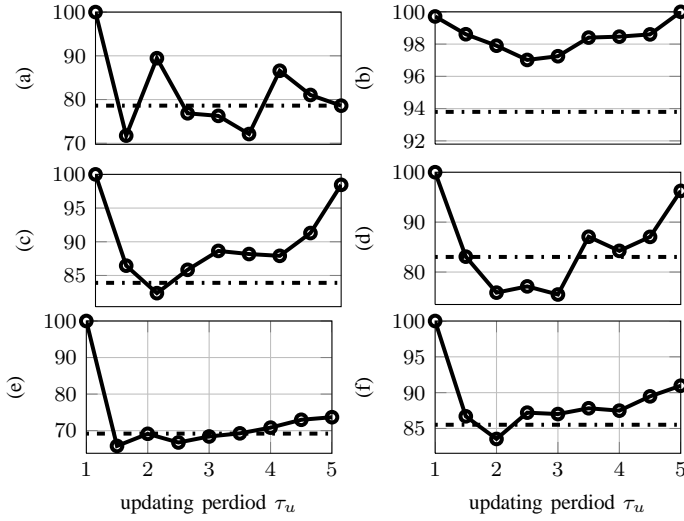


Fig. 11: Normalized cost (35)+(36) against updating period (consequently the number of iterations) for six different scenarios named (a) to (f). Solid lines represent the cost while dotted lines depict the obtained costs with the algorithm described in [3] for  $\delta = 2$

described in [3] will be implemented to show its efficiency on the cryogenic plant.

A six hour heat loads scenario presented by Fig. 12 will be divided in six one hour parts, to be simulated. Cost (35) + (36) defined in the previous section will be plotted against the chosen updating period. The result is presented by Fig. 11. It can be noted that the optimum updating period is different reading the scenario. It illustrates the fact that the updating period should be monitored to enhance performance.

The Fig. 11 also plots the obtained performance by monitoring the updating period using the algorithm [3]. It can be seen that it could lead to enhanced performances.

## VI. EXPERIMENTAL RESULTS

The control scheme derived in section IV-C and the solver depicted in section III-B has been implemented in the Schneider PLC described in section IV-D, in structured language. The objective of the section is triple. First, we want to show that the problem we derive in section IV-C is relevant regarding the control of a cryo-refrigerator submitted to transient heat loads. Then, we want to emphasize that the algorithm described in III-B is PLC compliant, event with polyhedral constraints. Finally, we will see that monitoring the updating scheme is very useful in this particular cases.

### A. Control result with real time PLC implementation

The plant has been submitted to a two hours scenario (first two hours of Fig. 12), starting from the equilibrium. The observed time per iteration is never longer than 500ms as expected and the problem preparation time do not exceed 500ms also. It allows the optimisation algorithm to iterate

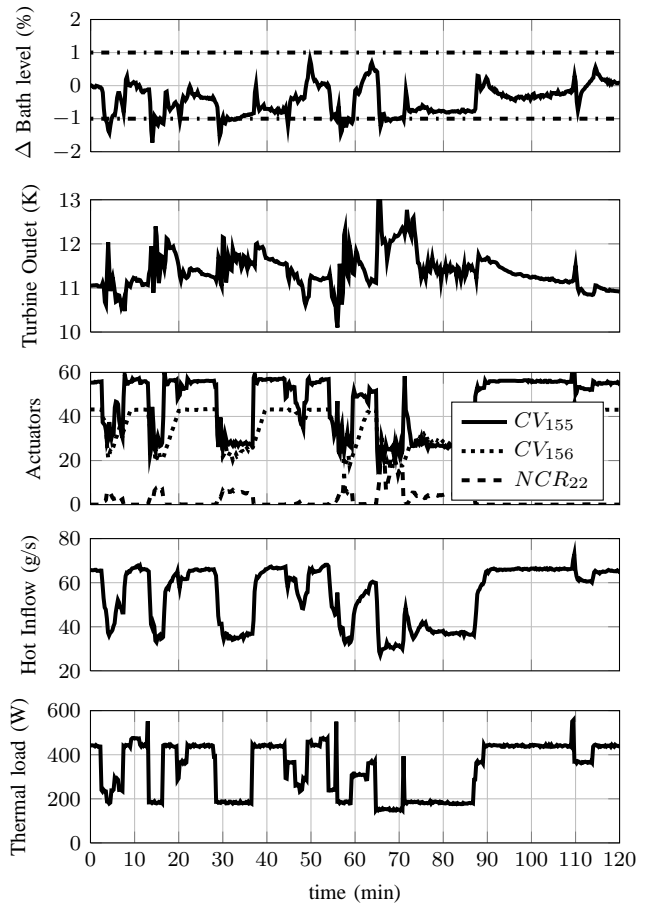


Fig. 13: Two hours heat load scenario. This Figure shows that the problem derived in section IV-C is relevant to control the plant. The  $\Delta$  level represent the helium level  $L1$  variation in the tank, Turbine stand for the output turbine temperature  $T_5$ . The Inflow depict the high pressure flow  $M_{12}$  coming in the cold-box

$4\tau_u - 1$  time. For the first test, we chose to use a  $\tau_u = 5s$  updating period. Fig. 13 shows that the control scheme is able to stabilize the plant and make the constraints to be respected, even if the pant is submitted to transient variable loads.

### B. Some leads on the updating scheme efficiency

The algorithm to update the updating period as been implemented on the PLC to show its efficiency. Unfortunately, the cost is not monitored but it is still possible to show result in the time domain. Fig 14 shows the difference between a constant updating period and a variable one. One can see that in the case of a serious change on the thermal load, the updating period is increasing to iterate more, while the algorithm is imposing a short updating period as soon as the problem is not changing much from an updating instant to another.

## VII. CONCLUSION

In this paper, an efficient way to control a cryogenic plant submitted to variable heat loads using an industrial PLC with

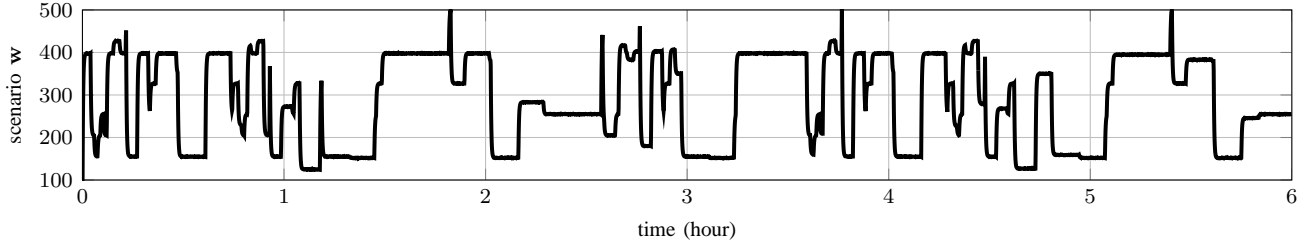


Fig. 12: Six hours heat loads scenario

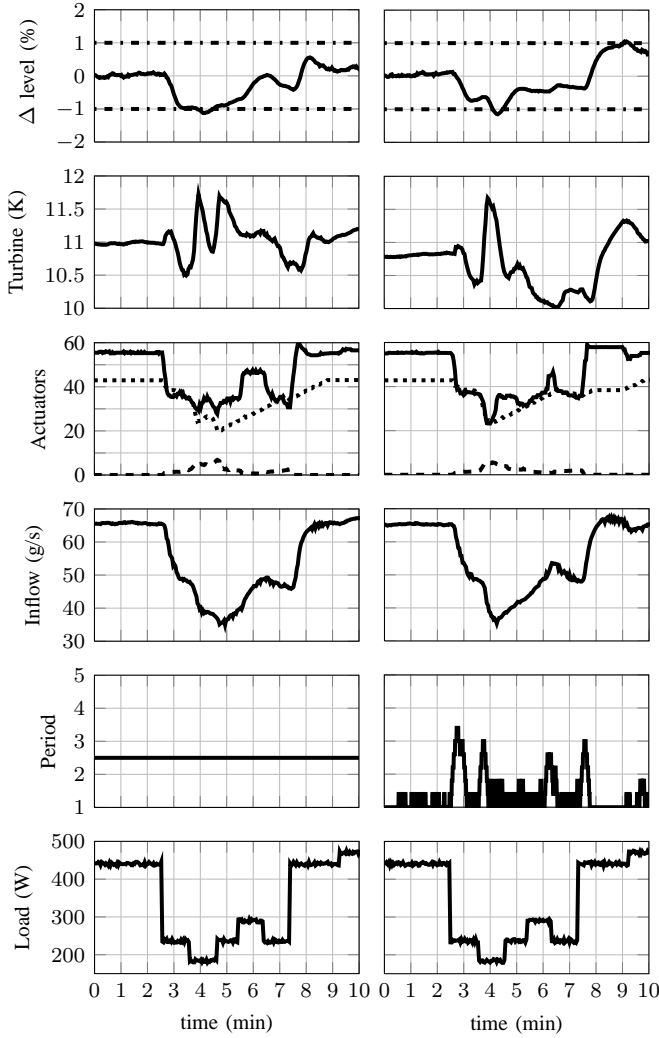


Fig. 14: Result with both constant (2.5s) and real-time updated updating period. It can be noticed that in the case of a heat load disturbance, the updating period is increasing (in order to make the number of iterations also increase), since the hot-starting solution is far from the actual solution. Period represents the updating period  $\tau_u$ . For actuators legend, please refer to Fig. 13.

reduced computing capabilities is proposed. It has been shown that in this application, an ODE-based solver gives robust sub-optimal solution even in the case where the hot-start is far from the optimal solution (in the case of an unpredicted disturbance for instance). The control scheme and the solver have been both validated experimentally.

Moreover an algorithm that automatically monitors the control updating period has been implemented and experimentally successfully tested.

Future investigation will aim at developing an MPC control scheme for a refrigerator submitted to multiple different thermal loads, including at 1.8K (super-fluid helium). Also, cryogenic systems could be very large (several buildings): the control scheme will be distributed in order to ensure a progressive integration to industrial systems.

#### ACKNOWLEDGMENT

Authors would like to thank every co-worker from the SBT for their kind help to improve models and control strategy and for their time to correct and discuss this paper. Authors give special thanks to Michel Bon-Mardion, Lionel Monteiro, François Millet, Christine Hoa, Bernard Rousset and Jean-Marc Poncet from SBT for their explanation about the process and their participation on experimental campaigns.

#### REFERENCES

- [1] M. Alami. *Stabilization of nonlinear systems using receding-horizon control schemes: A parameterized approach for fast systems*. Springer-Verlag, 2006.
- [2] M. Alami. A framework for monitoring control updating period in real-time NMPC schemes. In Lalo Magni, Davide Martino Raimondo, and Frank Allgöwer, editors, *Nonlinear Model Predictive Control*, volume 384 of *Lecture Notes in Control and Information Sciences*, pages 433–445. Springer Berlin Heidelberg, 2009.
- [3] M. Alami. Monitoring control updating period in fast gradient based NMPC. In *Control Conference (ECC), 2013 European*, pages 3621–3626, July 2013.
- [4] M. Alami. Fast NMPC, a reality-steered paradigm: Key properties of fast NMPC algorithms. In *Control Conference (ECC), 2014 European*, June 2014.
- [5] R. E. Bank, William M. Coughran Jr., Wolfgang Fichtner, Eric Grosse, Donald J. Rose, and R. Kent Smith. Transient simulation of silicon devices and circuits. *IEEE*

- Trans. on CAD of Integrated Circuits and Systems*, pages 436–451, 1985.
- [6] A. Bomporad and P. Patrinos. Simple and certifiable quadratic programming algorithms for embedded linear model predictive control. In *Proceeding of the IFAC Nonlinear Predictive Control Conference*, Noordwijkerhout, NL, 2012.
  - [7] F. Bonne, M. Alamir, and P. Bonnay. Physical control oriented model of large scale refrigerators to synthesize advanced control schemes. design, validation, and first control results. In *Proceedings of the Cryogenic Engineering Conference*, 2013.
  - [8] F. Bonne, M. Alamir, P. Bonnay, and B. Bradu. Model based multivariable controller for large scale compression stations. design and experimental validation on the lhc 18kw cryorefrigerator. In *Proceedings of the Cryogenic Engineering Conference*, 2013.
  - [9] François Bonne, Mazen Alamir, and Patrick Bonnay. Nonlinear observers of the thermal loads applied to the helium bath of a cryogenic Joule-Thompson Cycle. *Journal of Process Control*, 24(3):73–80, January 2014.
  - [10] F. Clavel. *Modélisation et contrôle d’un réfrigérateur cryogénique. Application à la station 800W à 4.5K du CEA Grenoble*. PhD thesis, EEATS, 2011.
  - [11] F. Clavel, M. Alamir, P. Bonnay, A. Barraud, G. Bornard, and C. Deschildre. Multivariable control architecture for a cryogenic test facility under high pulsed loads: Model derivation, control design and experimental validation. *Journal of Process Control*, 21(7):1030–1039, 2011.
  - [12] M. Diehl, H. G. Bock, and J. P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on control and optimization*, 43(5):1714–1736, 2005.
  - [13] M. Diehl, R. Findeisen, F. Allgower, H. G. Bock, and J. P. Schlöder. Nominal stability of real-time iteration scheme for nonlinear model predictive control. *IEE Proc.-Control Theory Appl.*, 152(3):296–308, 2005.
  - [14] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.
  - [15] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 2014. (in print).
  - [16] C. N. Jones, A. Domahidi, M. Morari, S. Richter, F. Ullmann, and M. Zeilinger. Fast predictive control: Real-time computation and certification. In *Proceeding of the IFAC Nonlinear Predictive Control Conference*, Noordwijkerhout, NL, 2012.
  - [17] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.
  - [18] Y. Nesterov. A method of solving a convex programming problem with convergence rate  $O(1/k^2)$ . *Soviet Mathematics Doklady*, 27(2):372–376, 1983.
  - [19] Schneider Electric, available at <http://www.schneider-electric.com/products/>. *Schneider PLC TSXP574634M Product data sheet*.
  - [20] V. M. Zavala, C. D. Laird, and L. T. Biegler. Fast implementation and rigorous models: can both be accommodated in NMPC? *International Journal of Robust and Nonlinear Control*, 18(8):800–815, 2008.